CS 6240 Parallel Data Processing With Map Reduce
Section 01 , HW-03 , Shabbir Saifee

Design Discussion

Page Rank Algorithm:

```
map (key, value)
        totalNodes = value from global counter
        Page node = get Page object from the value

        if node.PageRank == -1.0
                node.pageRank = 1.0/totalNodes;

        emiit(pageName,node)

        if node.adjacencyList is not null
                for page in adjacencyList
                        p = node.pageRank/ |adjacencylist |
                        emit(page,p)

reduce(key, <iterable>values)
        totalNodes = value from global counter
        delta = vale from global counter
        pageRank =0.0
        lambda = 0.85
        Page page
        if iteration == 1
                pageRank = 1.0/totalNodes

        else
                pageRank  = (1 – lambda)/totalNodes + lambda*delta/totalNodes
                for each node in values
                        if node is a neighbour
                                pageRank += lambda*node.pageRank
                        else
                                page.adjacencyList = node.adjacencyList

        if page.adjacencyList.size() == 0
                update delta counter value

        page.pageName = key
        page.pageRank = pageRank

        emit(key, page)
```

Pre-processing: This step retrieves all the out-link pages for a particular page removing self referencing nodes.

PageRank calculation: There are three ways we can calculate the delta (page rank sum of sink nodes) Method1. Create a job just to calculate sink nodes page rank sum and pass the calculated sum as a parameter to the MapReduce program to update page ranks. This will create overhead as we have to run an extra job for each iteration only to calculate delta.

Method 2: Implementing order inversion. Sending the previously calculated delta values to each reducer. This increases the data transfer between the mappers and reducers.

Method 3. Merging the computation of delta in the reduce phase. The reducer receives the previous delta value to calculate page ranks while calculating the delta value for the next iteration. So compared to method 1, we just use half the number of jobs and compared to method 2 instead of sending delta values across the reducers we only send it to a single reducer which limits the data transfer.

Thus, merging the delta computation into the reducer to use in the next iteration is the most optimal way.

Data Transferred RUN-1

Iteration	Data transferred from mapper to reducer in bytes	Data Transferred by reducer to HDFS(s3) in bytes
1	5337265487	1457408794
2	5337963154	1457413221
3	5338032145	1457409756
4	5337698431	1457411646
5	5337903124	1457416549
6	5337012756	1457406154
7	5337973675	1457409108
8	5337731170	1457416531

9	5337548012	1457415000
10	5338192112	1457416494

Data Transferred RUN-2

Iteration	Data transferred from mapper to reducer in bytes	Data Transferred by reducer to HDFS(s3) in bytes
1	5337964876	1457419430
2	5337931798	1457413749
3	5338097464	1457419564
4	5337847153	1457402148
5	5337931759	1457413675
6	5337976315	1457436942
7	5338154985	1457409642
8	5337931475	1457416145
9	5337697546	1457419877
10	5338106309	1457415896

Does the amount of data transferred in each iteration change over time?

The amount of data transferred almost remains **almost same** as it should because we are accessing same number of records in every iteration.
The slight change in data transfer is because of the change in the page rank at every iteration.

Top K pages

for top-k pages I have implemented secondary sort. To write top k pages override the run() method of reducer to run it 100 times only.

Composite key consists PageName and PageRank, and the key is sorted only by the pageRank in descending order.

```
public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{

        String line = value.toString();
        String pageName = line.substring(0,line.indexOf("[")-1);
        double pageRank = Double.parseDouble(line.substring(line.indexOf("{")          +1,
line.indexOf("}")));
        CompositeKey ck = new CompositeKey(pageName, pageRank);
        context.write(ck, NullWritable.get());

public void reduce(CompositeKey key, Iterable<NullWritable> values,
                Context context) throws IOException, InterruptedException {

        context.write(new Text(key.toString()), NullWritable.get());
    }
    // Override the run()

    @Override
    public void run(Context context) throws IOException, InterruptedException {
      setup(context);
      int count = 0;
      while(context.nextKey() && count++ < 100) {
         reduce(context.getCurrentKey(), context.getValues(), context);
      }

      cleanup(context);
    }
  }
```

Performance comparison

Run	Pre-processing time	Page rank calculation time	Top 100 pages
Run-1 time(seconds)	2741	3526	65

Run-2 time (seconds)	1494	2816	48

Evaluation and comparison of runtime results.
As expected the running time for RUN-2 is much better compared to the first one and the the simple reason is there are twice the number of machines in RUN-2 therefore dividing the work load over more number of machines results in faster execution.

Which computation phase showed good speedup?
From above observation it is clear that the Run-2 showed a better speed up. This is because of the fact that in RUN-2 we have 10 worker machines whereas as in RUN-1 we only have 5.
Speedup(preprocess) = 1.835
Speedup(PR calculation) = 1.25
Speedup(Top k) = 1.35

Page rank calculation speed up is less because since we have more machines in RUN-2 there will be more shuffling of data which slows down the execution process.

Output of simple-wikipedia data on local machine (top 100)

Page name	Page Rank
United_States_09d4	0.0064130687
Wikimedia_Commons_7b57	0.0044427285
Country	0.0039801501
England	0.0026984981
United_Kingdom_5ad7	0.0026626382
Europe	0.0026558634
Water	0.0026073551
Germany	0.0025934537
France	0.0025600957
Animal	0.0025244782
Earth	0.002455273
City	0.0024113113
Week	0.0020389185
Asia	0.0019565889
Sunday	0.0019023976
Monday	0.0018749525
Wednesday	0.0018561773
Money	0.0018390627
Friday	0.0018110686
Computer	0.0017994343
Plant	0.0017922745
Saturday	0.0017910602
Thursday	0.0017681686
English_language	0.001764345
Italy	0.0017597437
Tuesday	0.0017554577

India	0.0017315768
Wiktionary	0.0017293287
Government	0.0017267427
Number	0.0016105754
Spain	0.001595482
Japan	0.0015466867
Canada	0.0015281308
Day	0.0014905564
People	0.0014825704
Human	0.0014217605
China	0.0014048309
Australia	0.0013920633
Food	0.0013548671
Energy	0.0013263297
Sun	0.0013235991
Science	0.001312765
Mathematics	0.0013100166
Wikimedia_Foundation_83d9	0.0012431481
index	0.0012337915
Capital_(city)	0.0012154928
Russia	0.0012124797
Television	0.0011973029
State	0.0011909641
Music	0.0011865048
Year	0.0011626982
Greece	0.0011404373
Wikipedia	0.0011367624
Language	0.0011151825
Metal	0.0011126382
Scotland	0.0011032631
2004	0.0010931601
Sound	0.0010677311
Religion	0.0010533434
Planet	0.0010423836
Greek_language	0.0010340736
London	0.0010274227
Africa	0.0009894055
20th_century	0.000980858
19th_century	0.0009583154
World	0.0009525826
Poland	0.0009439968
Society	0.0009349541
Law	0.000932523
Geography	0.0009306543
Latin	0.0009179194
Liquid	0.0009067753
Scientist	0.0008933636
Sweden	0.0008905826
History	0.00088995

Atom	0.0008878242
Netherlands	0.0008800744
Culture	0.0008743693
War	0.0008695992
Light	0.0008569147
Building	0.0008516468
Turkey	0.0008443815
God	0.0008434277
Plural	0.0008384918
Information	0.0008384513
Centuries	0.0008295316
Portugal	0.0008106974
Inhabitant	0.0008029712
Capital_city	0.0007939635
Denmark	0.0007918159
Austria	0.0007912381
Chemical_element	0.000789651
Cyprus	0.0007767629
Ocean	0.0007740563
North_America_e7c4	0.0007714033
Electricity	0.0007592913
Biology	0.0007580224
Image	0.0007517421
University	0.0007491342
Species	0.0007491077

Output of full-wikipedia data on aws (top 100)

Page Name	**Page Rank**
United_States_09d4	0.0029432651
2006	0.0026295337
United_Kingdom_5ad7	0.001398654
2005	0.0012119063
Biography	0.0009537568
Canada	0.0009144504
England	0.0009076611
France	0.0008964338
2004	0.0008432845
Germany	0.0007708756
Australia	0.0007479867
Geographic_coordinate_system	0.0007203728
2003	0.0006793393
India	0.0006582361
Japan	0.0006535736
Italy	0.0005462764
2001	0.0005450996
2002	0.0005388694

Internet_Movie_Database_7ea7	0.0005332917
Europe	0.000519088
2000	0.0005102238
World_War_II_d045	0.0004921589
London	0.0004751619
1999	0.0004511146
Population_density	0.0004508397
Record_label	0.0004486688
English_language	0.000447697
Spain	0.0004467864
Russia	0.0004222001
Race_(United_States_Census)_a07d	0.000417375
Wiktionary	0.0004092666
1998	0.0003898918
Wikimedia_Commons_7b57	0.0003886231
Music_genre	0.00038034
1997	0.0003718744
Scotland	0.0003660813
New_York_City_1428	0.0003659563
Football_(soccer)	0.0003553014
1996	0.0003489945
Sweden	0.0003435718
Television	0.0003433874
Census	0.0003293225
1995	0.0003286431
California	0.0003265592
China	0.0003218821
Square_mile	0.000321111
Netherlands	0.0003170511
New_Zealand_2311	0.0003162375
1994	0.0003136732
1991	0.0002991882
1993	0.0002965263
1990	0.0002947089
New_York_3da4	0.0002931179
Public_domain	0.0002917087
1992	0.0002842992
Film	0.0002820836
Actor	0.000278932
Scientific_classification	0.0002783588
United_States_Census_Bureau_2c85	0.0002782954
Norway	0.000276569
Ireland	0.0002755738
Poland	0.0002724156
Population	0.0002720893
1989	0.0002664819
1980	0.0002602149
January_1	0.000259943
Marriage	0.000258213

Brazil	0.0002577374
Mexico	0.0002567252
Latin	0.0002559888
1986	0.0002531556
Politician	0.0002510109
1985	0.0002469753
1979	0.0002465816
1982	0.0002459481
French_language	0.0002457838
1981	0.0002457424
1974	0.0002435091
Switzerland	0.0002413299
South_Africa_1287	0.0002412957
1984	0.0002412832
Album	0.0002411106
1987	0.0002410744
1983	0.0002409567
Per_capita_income	0.0002398727
1970	0.0002370693
Record_producer	0.0002365992
1988	0.0002355734
1976	0.0002343235
Km²	0.0002319717
1975	0.0002316079
Paris	0.0002285341
1969	0.0002284329
Greece	0.0002278338
1972	0.0002269765
1945	0.0002269434
1977	0.0002250859
1978	0.0002239819
Soviet_Union_ad1f	0.0002239648
Personal_name	0.0002232998

By looking at the final page ranks and observing the links in the pages, there are many pages that just point towards one another. Like page 2000 refers to all the pages from the same decade and vice versa. So, therefore these pages are just referencing to one another. Which is kind of like self-reference. This is not a great way to go about page rank.